SafeStr

Daniel Plakosh, Software Engineering Institute [vita[1]]

2005-09-27; Updated 2008-07-17

L1 / D/P, L[2]

The C String Library (SafeStr) from Messier and Viega provides a rich string-handling library for C that has secure semantics yet is interoperable with legacy library code in a straightforward manner.

Development Context

String manipulation

Technology Context

C, UNIX, Win32

Attacks

Attacker executes arbitrary code on machine with permissions of compromised process or changes the behavior of the program.

Risk

Standard C string manipulation functions are prone to programmer mistakes that can result in buffer overflow vulnerabilities.

Description

The C String Library (SafeStr) from Messier and Viega provides a rich string-handling library for C that has secure semantics yet is interoperable with legacy library code in a straightforward manner [Messier 03].

The SafeStr library uses a dynamic approach for C that automatically resizes strings as required. SafeStr accomplishes this by reallocating memory and moving the contents of the string whenever an operation requires that a string grow in size. As a result, buffer overflows should not result from using the library.

The SafeStr library is built around the safestr_t type. The safestr_t type is compatible with char * and allows safestr_t structures to be cast as char * and behave as C-style strings. The safestr_t type keeps accounting information (e.g., the actual and allocated length) in memory directly preceding the memory referenced by the pointer.

The SafeStr library supports immutable strings. Strings can be specified as immutable during initialization or by calling

```
void safestr_makereadonly(safestr_t s);
```

Immutable strings cannot be modified using the SafeStr API. However, the memory can still be overwritten. The library only prevents writes initiated through SafeStr functions.

The SafeStr API can help track trusted and untrusted data in the style of Perl’s taint mode. A developer can use this mechanism to mark strings originating from untrusted sources as such. Strings that have been checked for potentially malicious input could subsequently be marked as trusted. When modifying a string, the trusted property of that string is set to "untrusted" if any of the operands are untrusted. When creating

1. http://buildsecurityin.us-cert.gov/bsi/about_us/authors/268-BSI.html (Plakosh, Daniel)

a new string from operations on other strings, the new string is marked as trusted only if all the strings that influence its value are trusted.

The trust property will not properly propagate if the SafeStr API is circumvented. The SafeStr API does not currently provide any routines that check the trusted flag. However, you can explicitly check the flag yourself as shown in Figure 1.

Figure 1. Trusted and untrusted data in SafeStr

```
1. int safer_system(safestr_t cmd) {
2.   if (!safestr_istrusted(cmd)) {
3.     printf("Untrusted data in safer_system!\n");
4.     abort();
5.   }
6.   return system((char *)cmd);
7. }
```

Error handling in SafeStr is performed using XXL[22], a library that provides both exceptions and asset management for C and C++. The caller is responsible for handling exceptions thrown by SafeStr and XXL. If no exception handler is specified, the default action is to output a message to stderr and call `abort()`. The dependency on XXL can be an issue because both libraries need to be adopted to support this solution.

SafeStr is released under an open source BSD-style license.

References

[ISO/IEC 99]	ISO/IEC. *ISO/IEC 9899 Second edition 1999-12-01 Programming languages — C*. International Organization for Standardization, 1999.
[Messier 03]	Messier, Matt & Viega, John. *Safe C String Library v1.0.3*. http://www.zork.org/safestr (2005).

Pearson Education, Inc. Copyright

22. http://www.zork.org/xxl